Composing the Transmission File and Return Data For the Modernized e-File System An Overview

Release No: 3.0
Date: August 31, 2004

Composing the Transmission File and the Return Data For the Modernized e-File System An Overview

Revision History

Revision Number	Revision Date	Summary of Changes	Changes marked
1.0 Draft	09/25/02	Release 1.0 Draft	
2.0 Draft	04/30/03	Release 1.0 Draft – generalized to include 99x returns	
3.0 Draft	07/31/03	Release 1.0 Draft – editorial changes	
4.0 Draft	09/30/03	Release 1.0 Draft – updated file structure and schema changes	
5.0 Draft	08/31/04	Release 3.0 Draft – updated Section 8.3 for binary attachment reference requirement. Plus, added new section 9 for "Guidelines for composing the return data structure"	

Table of Contents

1. Introduction

This document is intended for the software developers who compose the tax return data in XML format for the Modernized e-File System, and for electronic transmitters who compose the transmission file that is submitted to the IRS. It establishes the structure of the transmission file, and general structure of the tax returns that are supported by the Modernized eFile system.

It includes an example Consolidated 1120 return and shows how it is enclosed within the Transmission File. It documents general considerations about using the XML schemas and provides guidance to the EROs (Electronic Return Originators) on composing the return data (structure), and to the transmitters on composing the transmission file (structure) for submission to the IRS.

This document is in draft stage and will be embellished over time.

The user's cooperation is requested in developing a quality document that provides guidance on the use of XML Schemas to compose the return data and transmission files. If you notice errors (typographical, technical, or usage) or if you have any suggestions and/or comments, please let us know. Please email your comments to 1120@irs.gov

2. Transmission File Structure

The transmission file is a MIME (Multipurpose Internet Mail Extensions) multi-part document that conforms to "SOAP 1.1 with attachments" standard. The first part of the multi-part document is the SOAP envelope and remaining parts are SOAP attachments. MIME boundaries separate the parts in the multi-part document. The SOAP envelope contains transmission level information, and each SOAP attachment is a tax return (or an extension).

The SOAP envelope consists of a SOAP header and a SOAP body. The SOAP header, also referred to as the *transmission header* in the Modernized e-File system, contains information about the transmitter and the transmission. The SOAP body, also referred to as the *transmission manifest*, contains a list of all SOAP attachments (tax returns and/or extensions) in the transmission file.

Each SOAP attachment in the transmission file is a MIME multi-part document that contains data for either a tax return or an extension. The return consists of one or more MIME parts. The first part is always the return data, and the remaining parts, if any, contain non-XML information, also referred to as “binary attachments”.

The structure of the transmission files is depicted below- first graphically; followed by the structure using MIME header fields.

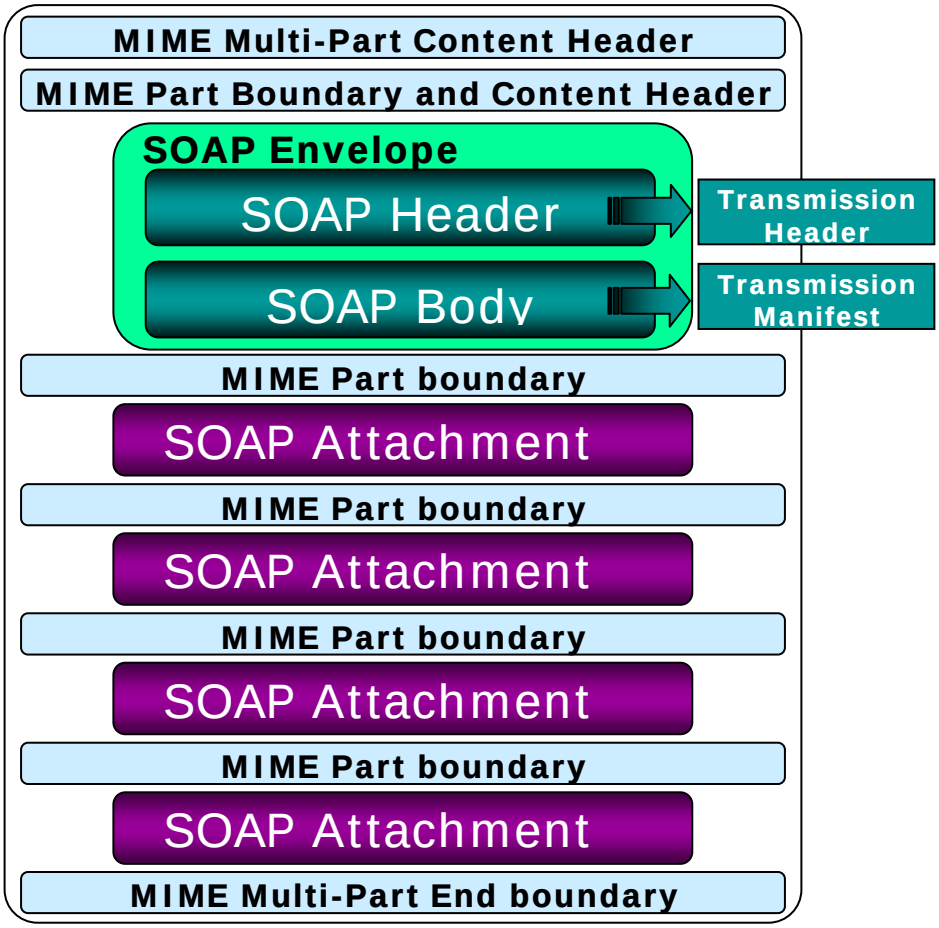


Figure 1 – Graphic representation of a Transmission File structure

The structure of the transmission file using MIME header fields is shown below followed by explanations of the header fields and whether or not they are required. The text to the right side of the brackets explains the purpose of the header fields. Note that the data (structure) for the first return in the transmission file is indented for easy reading, is in bold text and in blue color.

Note that the values in brackets e.g., "<MIME1120Boundary>" need to be replaced by the application composing the file.

Structure of the transmission file:

MIME-Version: 1.0	required verbatim
Content-Type: Multipart/Related; boundary=<MIME1120Boundary>; type=text/xml	*Multi-part content header*
Content-Description: This is a sample structure of a transmission file.	- description header
X-eFileRoutingCode: MEF	- custom e-File header
--<MIME1120Boundary>	boundary, separates parts
Content-Type: text/xml; charset=UTF-8	*body part content header*
Content-Transfer-Encoding: 8bit	for the SOAP envelope
Content-Location: <Envelope1120>	

(…SOAP Envelope goes here… contains a SOAP Header and a SOAP body)

--<MIME1120Boundary>	**2nd part, a return, starts here**
Content-Type: Multipart/Related; boundary=<Return001PartBoundary>; type=text/xml	
Content-Location: <01000020030860000001>	**same as the ReturnID**
Content-Description: data for the first return (one or more parts) follows.	**Optional description**
--<Return001PartBoundary>	**boundary, separates parts**
Content-Type: text/xml; charset=UTF-8	***body part headers***
Content-Transfer-Encoding: 8bit	**for the return**
Content-Location: <ReturnData001>	
(Return (XML) data for the return goes here)	**parent return data**
--<Return001PartBoundary>	**separates parts in return**
Content-Type: application/pdf	**body part header**
Content-Transfer-Encoding: Binary	
Content-Location: <BinaryAttachment001>	
Content-Description: 8453 Signature Document	**Title of non-XML doc**
(Binary attachment file goes here)	**PDF file**
(… other non-XML (binary) attachments , if any, go here)	
--<Return001PartBoundary>--	**end of return001**
--<MIME1120Boundary>	Begin 2nd return
Content-Type: Multipart/Related; boundary=<Return002PartBoundary>; type=text/xml	Multi-part header for
Content-Location: <01000020020860000002>	the second return in
Content-Description: This is the second return in the transmission file.	the transmission file
--<Return002PartBoundary>	all data (parts) of second return go here

(…other parts in the return, if any, go here)

--<Return002PartBoundary>-- ⎫ end of 2nd return

--<MIME1120Boundary> ⎫ Begin 3rd part (return)

(… other returns go here…)

--<MIME1120Boundary>-- ⎫ end of parts in trans file

2.1. Header fields used in the transmission file

The *MIME content headers,* or simply the *content headers* are used to describe the contents of MIME parts. The *multi-part content header* (Content-Type="Multipart/Related") specifies a boundary value that separates MIME parts in a MIME multi-part structure. Other header fields provide additional information about the MIME part. There is one *multi-part content header* for each MIME multi-part structure (the transmission file, and each return).

Since the Transmission file is a multi-part structure, there is one *multi-part content header* for the transmission file. The value in the *boundary* parameter (<MIME1120Boundary> in the transmission file structure above) of this content header separates parts in the transmission file. The first part is the SOAP envelope, and all other parts are SOAP attachments (returns or extensions).

Since each tax return is also a MIME multi-part structure, it contains one *multi-part content header*. The value in the *boundary* parameter (<Return001PartBoundary> for the first return in the transmission file structure above) of this content header separates parts (return XML data and non-XML data) that make up the tax return or the extension.

2.1.1 MIME-Version header field

The MIME-Version header field is **required** at the top level (of a message) in the transmission file. It is not required for each body part of a multipart document. It indicates that the transmission file conforms to MIME version 1.0 standard. It must be included with the following verbatim text:

MIME-Version: 1.0

2.1.2 Content-Type header field

The Content-Type header field is **required** for each MIME part (transmission envelope, return, extension, and each part in the return/extension). The Content-Type header field describes the nature of the data in the MIME part by giving media type and subtype identifiers.

The value of the Content-Type header field for a multi-part message must be set to "Multipart/Related" and a value must be provided for both the *boundary* parameter and the *type* parameter. It must appear as follows:

Content-Type: Multipart/Related; boundary=<MIME1120Boundary>; type=text/xml

The *boundary* parameter separates parts in the multi-part document- it separates the transmission envelope and the tax returns/extensions in a transmission file, and separates return (XML) data from the non-XML attachments in a tax return/extension. The value for the *boundary* parameter ("<MIME1120Boundary>" above) is to be created by the application composing the transmission file. The *type* parameter indicates the content type of the multipart/related object. It must have the value "text/xml" for the transmission envelope part as per SOAP 1.1 specification.

Note that the value of a parameter does not include the quotes that are used when describing the value. That is, the quotation marks in a quoted-string are not a part of the value of the parameter. So, the following two forms are equivalent:

Content-Type: text/xml; charset=UTF-8

Content-Type: “text/xml”; charset=UTF-8

The value of the Content-Type header field is set to either "text/xml" when describing an XML part, or to "application/pdf" when describing a non-XML part in a multi-part structure.

When the Content-Type header field is included in the *body part header* of the SOAP envelope (in the transmission file), or the return data (in the return), its value must be set to "text/xml" as shown below:

Content-Type: text/xml; charset=UTF-8

When Content-Type header field is included in the *body part header* of a 'binary attachment', its value must be set to "application/pdf" as shown below:

Content-Type: application/pdf

The *charset* parameter must be set to the value "UTF-8" when the value of the header is "text/xml". This parameter is not required when the value of the header is "application/pdf".

The names of the parameters and the type and subtype values are not case sensitive, however, the parameter values are. For example, “text/xml” and “Text/xml” are equivalent. However, the value for the *boundary* parameter (in the Multi-part content header) is case-sensitive. Hence, the boundary=MIME1120Boundary, and boundary=Mime1120Boundary are not equivalent.

2.1.3 Content-Description header field
The Content-Description header field is **required** for a MIME part that contains a non-XML binary attachment (type=”application/pdf”) in a return and in an extension. Its value must be the title of the binary attachment. The title has been fixed for some binary attachments. For example, the title of the signature binary attachment is fixed to “8453 Signature Document”. The filer provides a title for binary attachments whose title has not been fixed by the IRS. For example, if the binary attachment contains a scanned image of the organization chart, the title may be “Organization Chart for XYZ organization for year 2001”.

The Content-Description header field is **optional** for all other types (type=”text/xml”, text=”Multipart/Related”) of MIME parts. If present, it is not processed by the Modernized e-File system.

2.1.4 X-eFileRoutingCode header field
The X-eFileRoutingCode header field is **required** in the message header of the transmission file. It is a custom (not a standard MIME header) header field created specifically for use by the Modernized e-File system. Its purpose is to indicate the ‘kind’ of data included in the transmission file. There are three kinds of data that can potentially be included in a transmission file.

A transmission file can contain **only one** of the three kinds of data. For the Modernized e-File system, only the first entry “MEF” is allowed. The other two labels are used for 94X processing. The three kinds are:

1) When the transmission file contains Corporate tax returns (1120, 1120S), and/or TEGE returns (990,990-EZ,1120-POL) and/or the F8868 Extensions, the value of the X-eFileRoutingCode header field must be set to “MEF” as follows:

X-eFileRoutingCode: MEF

2) When the transmission file contains 94x family of tax returns (940, 940PR, 941, 941PR, 941SS), the value of the X-eFileRoutingCode header field must be set to “94X” as follows:

X-eFileRoutingCode: 94X

3) When the transmission file contains data for Pin Registration, the value of the X-eFileRoutingCode header field must be set to “PINREGISTRATION” as follows:

X-eFileRoutingCode: PINREGISTRATION

2.1.5 Content-Transfer-Encoding header field
The Content-Transfer-Encoding header field is **required** for MIME parts whose Content-Type header has the value "text/xml" or "application/pdf". Its value must be set to "8Bit" for MIME parts of Content-Type "text/xml", and it must be set to "Binary" for MIME parts of Content-Type "application/pdf". These values indicate that no encoding has been applied to the body part. The value for this header must be specified as follows:

1) The Content-Transfer-Encoding header for the MIME part that contains the XML data must be:

Content-Transfer-Encoding: 8Bit

2) The Content-Transfer-Encoding header for the MIME part that contains the binary attachment must be:

Content-Transfer-Encoding: Binary

3) The Content-Transfer-Encoding header field is not required when the Content-Type header field has any other value e.g., "Multipart/Related". However, If it is included with the *multi-part content header* (Content-Type set to "Multipart/Related"), it is not permitted to have any value other than "7Bit", "8Bit", or "Binary".

2.1.6 Content-Location header field
The Content-Location header field is **required**. It specifies an URI that labels the content of the body part in whose heading it is placed. References are made to these labels from the root body part of the multi-part/related MIME message.

In the transmission file, there are references to these labeled body parts (returns) from the transmission manifest (the root body part).

In the return, also a multi-part/related MIME structure, there are references to the labeled body parts (binary attachments) from inside the return structure.

The URIs in the Content-Location headers SHOULD be globally unique. The Modernized e-File system requires that the URIs within the transmission file be unique i.e., each return have a unique ReturnID (which is its Content-Location), AND, the URIs within the return be unique i.e., each MIME part within a return have a unique value for the Content-Location header. It does not require that Content-Location values inside a return be unique within the transmission file; they just need to be unique within the return.

3. What does the transmitter do when composing the transmission file?

The transmitter is an IRS authorized *e-file* provider that transmits transmission files to the IRS. The transmitters receive the electronic data for the returns from the EROs (Electronic Return Originators). This section describes the structure of the transmission file that the transmitter must follow and general guidance on composing the transmission file.

The transmission file is a MIME multi-part structure that conforms to the "SOAP 1.1 with attachments" standard. The transmission file contains two kinds of parts- the SOAP envelope (also referred to as the *transmission envelope*), and SOAP attachments (a return or an extension). MIME boundaries separate the parts in the multi-part MIME structure. The SOAP envelope, an XML structure, maintains transmission level information, and each SOAP attachment is a return or an extension.

The transmitter composes the SOAP envelope, and the transmission file. The ERO composes SOAP attachments (the returns and/or extensions) and provides to the transmitter for inclusion in the transmission file.

The SOAP envelope consists of a SOAP header and a SOAP body. The SOAP header, also referred to as the *transmission header* in the Modernized e-File system, contains information about the transmitter and the transmission. The SOAP body, also referred to as the *transmission manifest*, contains a list of all SOAP attachments in the transmission file.

The transmitter needs to know the ReturnID of each return that is included in the transmission file. The ERO provides this ID along with the return/extension data structure of each return/extension to the transmitter. The transmitter uses the ID to create the transmission manifest that enumerates (and points to) each return/extension in the transmission file. The contentLocation attribute of the *Reference* element in the transmission manifest is set to the *Content-Location* header field value of the return.

Each SOAP attachment, a MIME part in the transmission file, is itself a MIME multi-part structure that contains data for a return or an extension.

The general structure of the transmission file, a MIME multi-part structure, is depicted below. It is followed by explanations where appropriate. Note that the values in angled brackets e.g., "<Return001PartBoundary>" need to be replaced by the application composing the MIME structure for the return.

MIME-Version: 1.0 — required verbatim
Content-Type: Multipart/Related; boundary=<MIME1120Boundary>; type=text/xml — Multi-part header
Content-Description: This is a sample structure of a transmission file. — Description header
X-eFileRoutingCode: MEF — custom e-File header

--<MIME1120Boundary> — boundary, separates parts
Content-Type: text/xml; charset=UTF-8 — *body part header*
Content-Transfer-Encoding: 8bit — for the SOAP envelope
Content-Location: <Envelope1120>

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns=http://www.irs.gov/efile
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:efile="http://www.irs.gov/efile"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/  ../message/SOAP.xsd
http://www.irs.gov/efile ../message/efileMessage.xsd">
      <SOAP:Header>
            <efile:TransmissionHeader transmissionVersion="2002V01">
                  <TransmissionId>TID01</TransmissionId>
                  <Timestamp>2003-05-01T12:11:17+05:01</Timestamp>
                  <Transmitter>
                        <ETIN>12345</ETIN>
                  </Transmitter>
            <ProcessType>T</ProcessType>
            </efile:TransmissionHeader>
      </SOAP:Header>
      <SOAP:Body>
            <efile:TransmissionManifest count="1">
            <Reference contentLocation="99009920031210123456"
                  electronicPostmark="2002-03-27T08:20:00-05:00" >
                  </Reference>
            </efile:TransmissionManifest>
      </SOAP:Body>
</SOAP:Envelope>
```

SOAP Header also called Transmission Header
SOAP Body also called Transmission Manifest
SOAP Envelope

--<MIME1120Boundary> — separates returns in trans file

(return data, composed by the ERO, goes here) — return data for 1st return

--<MIME1120Boundary>-- — end of parts in trans file

Transmission file showing the SOAP Envelope structure

3.1. MIME content headers and MIME parts in the transmission file

The *MIME content headers,* or simply the *content headers* are used to describe the contents of MIME parts. For an explanation of the header fields and whether or not they are required, please see the Chapter titled "Transmission File Structure".

The transmission file contains two kinds of MIME parts- the SOAP envelope (also referred to as the *transmission envelope*), and SOAP attachments (a return or an extension). The transmitter composes the SOAP envelope. The ERO composes SOAP attachments (the returns and extensions) and provides to the transmitter for inclusion in the transmission file.

The SOAP envelope, an XML document, consists of two elements- a transmission header and a transmission manifest. The transmission header contains information about the overall transmission. The transmission manifest is a list of all returns included in the transmission file. The location (the **Content-Location** header field value) of each return/extension is specified in the transmission manifest.

The first MIME part in the transmission file
The first MIME part in the transmission file is always the SOAP envelope. The SOAP envelope is an XML document that conforms to the SOAP schema described in the soap.xsd file. It is composed by the transmitter.

The MIME parts 2..n in the transmission file
The MIME parts 2..n each represent a return or an extension. These MIME parts (returns or extensions) are composed by the ERO and provided to the transmitter for inclusion in the transmission file. The MIME parts are separated by the boundary value created by the transmitter and specified by the *boundary* parameter in the Content-Type header field of the entire message.

4. What does the ERO do when composing the return/extension data?

The ERO (Electronic Return Originator) is the authorized Modernized e-File provider that originates the electronic submission of a return/extension using approved software. The EROs may originate the electronic submission of returns or extensions they have prepared and/or collected from taxpayers. This section describes the structure of an electronic return that the ERO must follow when composing the return data.

The electronic return is a MIME multi-part structure. The ERO is required to create the return data per the XML Schema for the return type (1120, 1120S, 990, etc.), and enclose the return data into the MIME structure. The return data can be made up of one or more MIME parts. The first MIME part always contains the return (XML) data, and subsequent parts, if present, contain non-XML (PDF) data. In a consolidated return, the consolidated data, data for the parent corporation, and data for all subsidiaries is included in the first MIME part, and each non-XML data (PDF) file is enclosed into its own MIME part.

The general structure of the return is depicted below- first graphically; followed by the structure of the return using MIME header fields.

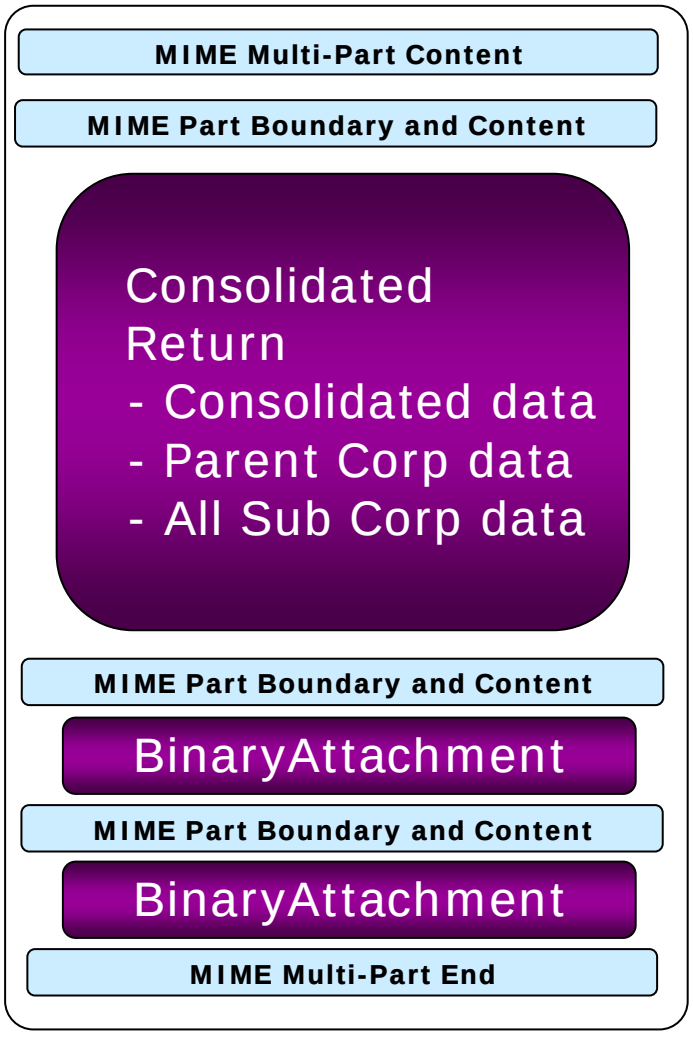


Figure 2 – Graphic representation of a Return structure

The structure of the return is depicted below using MIME header fields followed by explanations where appropriate. Note that the values in angled brackets e.g., "<Return001PartBoundary>" need to be replaced by the application composing the MIME structure for the return.

Content-Type: Multipart/Related; boundary=<Return001PartBoundary>; type=text/xml Content-Location: 01000020020860000001 Content-Description: Consolidated Return for XYZ corporation.	Multi-part content header for the return
--<Return001PartBoundary> Content-Type: text/xml; charset=UTF-8 (parent corp's return) Content-Transfer-Encoding: 8bit Content-Location: <ReturnData001>	separates parts in the return *body part header* for the **parent** corporation's return
<?xml version="1.0" encoding="UTF-8"?> <Return xmlns=http://www.irs.gov/efile ….returnVersion="2002V01"> <ReturnHeader binaryAttachmentCount="1" subsidiaryReturnCount="1"> (header data goes here) </ReturnHeader> <ReturnData > (Consolidated data for the return goes here) </ReturnData> <ParentReturn> (data for the parent return goes here) </ParentReturn> <SubsidiaryReturn> (data for subsidiary1 return goes here) </SubsidiaryReturn> </Return>	Consolidated Return includes - Consolidated data - parent corp data - subsidiary data
--<Return001PartBoundary> Content-Type: **application/pdf** Content-Transfer-Encoding: Binary Content-Location: <BinaryAttachment001> Content-Description: 8453 Signature Document	separates parts in return body part header for the **non-XML** data contains Signature document
(Binary attachment for signature goes here)	PDF file
--<Return001PartBoundary>--	end return 01000020020860000001

General structure of a return, a MIME multi-part structure

4.1. MIME headers and MIME parts in the return/extension

The *MIME content headers,* or simply the *content headers* are used to describe the contents of MIME parts. For an explanation of the header fields and whether or not they are required, please see the Chapter titled "Transmission File Structure".

The first MIME part in the return/extension
The first MIME part in the return is always the return or the extension data (content-type="text/xml"). The return/extension data is provided as an XML (instance) document. Please refer to the XML Schemas for the various return types (1120, 1120S, 990, …) and the extensions (8868) that can be filed electronically.

The MIME parts 2..n in the return/extension
MIME parts 2..n, if present, contain non-XML (binary) data. The binary data must be in PDF format. The Content-Type header field for the MIME part must be set to “application/pdf”.

5. A sample transmission file containing a Consolidated 1120 tax return

Here is a sample transmission file that contains the structure of one consolidated 1120 tax return. In practice, however, the file will most likely contain multiple returns/extensions. Please see the instance document, Example_TransmissionWithConsolidatedReturn.txt, included with the schemas for corporate returns.

MIME-Version: 1.0 — required verbatim

Content-Type: Multipart/Related; boundary=<MIME1120Boundary>; type=text/xml
Content-Description: Sample structure of a transmission file containing ONE return structure
X-eFileRoutingCode: MEF
— Message header of the transmission file

--<MIME1120Boundary>
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Location: <Envelope1120>
— separates parts in trans file; *body part header* for the SOAP envelope

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns=http://www.irs.gov/efile
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:efile="http://www.irs.gov/efile"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/  ../message/SOAP.xsd
http://www.irs.gov/efile ../message/efileMessage.xsd">
      <SOAP:Header>
            <efile:TransmissionHeader transmissionVersion="2002V01">
                  <TransmissionId>TID01</TransmissionId>
                  <Timestamp>2003-05-01T12:11:17+05:01</Timestamp>
                  <Transmitter>
                        <ETIN>12345</ETIN>
                  </Transmitter>
            <ProcessType>T</ProcessType>
            </efile:TransmissionHeader>
      </SOAP:Header>
      <SOAP:Body>
            <efile:TransmissionManifest count="1">
            <Reference contentLocation="990099200312101234567"
                  electronicPostmark="2003-04-30T08:20:00-05:00" >
                  </Reference>
            </efile:TransmissionManifest>
      </SOAP:Body>
</SOAP:Envelope>
```

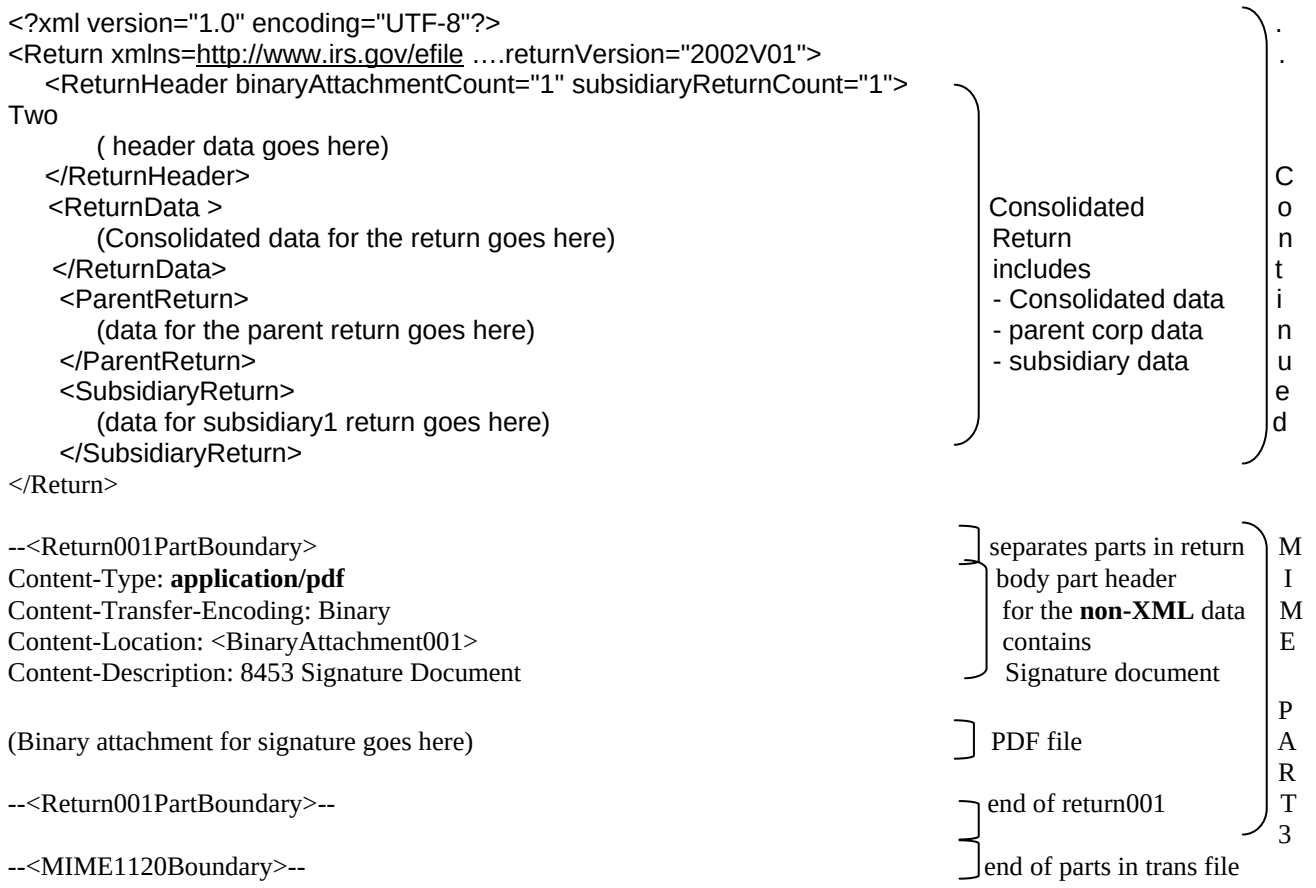
SOAP Header also called Transmission Header; SOAP Body also called Transmission Manifest

SOAP MIME Part 1

--<MIME1120Boundary>
Content-Type: Multipart/Related; boundary=<Return001PartBoundary>; type=text/xml
Content-Location: 99009920031210123456
Content-Description: Consolidated Return for XYZ corporation.
— separates parts in trans file; Message header of the return

--<Return001PartBoundary>
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Location: <ReturnData001>
— separates return parts; *body part header* for the return data (XML)

MIME Part 2.


<?xml version="1.0" encoding="UTF-8"?>
<Return xmlns=http://www.irs.gov/efile ….returnVersion="2002V01">
<ReturnHeader binaryAttachmentCount="1" subsidiaryReturnCount="1">
Two
(header data goes here)
</ReturnHeader>
<ReturnData >
(Consolidated data for the return goes here)
</ReturnData>
<ParentReturn>
(data for the parent return goes here)
</ParentReturn>
<SubsidiaryReturn>
(data for subsidiary1 return goes here)
</SubsidiaryReturn>
</Return>
Consolidated Return includes
- Consolidated data
- parent corp data
- subsidiary data
Consolidated Return
--<Return001PartBoundary>
Content-Type: application/pdf
Content-Transfer-Encoding: Binary
Content-Location: <BinaryAttachment001>
Content-Description: 8453 Signature Document
(Binary attachment for signature goes here)
--<Return001PartBoundary>--
--<MIME1120Boundary>--
separates parts in return
body part header for the non-XML data contains Signature document
PDF file
end of return001
end of parts in trans file
MIME PART 3

6. General philosophy on data elements in Schemas

In general, most data elements in the schemas for each form, schedule, and supporting document have been declared optional. Most of the required elements are in the schema for the return header. The schema for the return header contains identifying information about the entity filing the return, the officer responsible for the data in the return, the preparer, and the preparing firm. Hence there are very few data elements for which you must provide data values. This philosophy of keeping most data elements optional in the schemas is consistent with the way paper returns are filed i.e., the taxpayer and return preparer have the responsibility to provide information as specified by IRS forms, instructions, and regulations.

7. How are the schema files physically organized (i.e., what's in each file)?

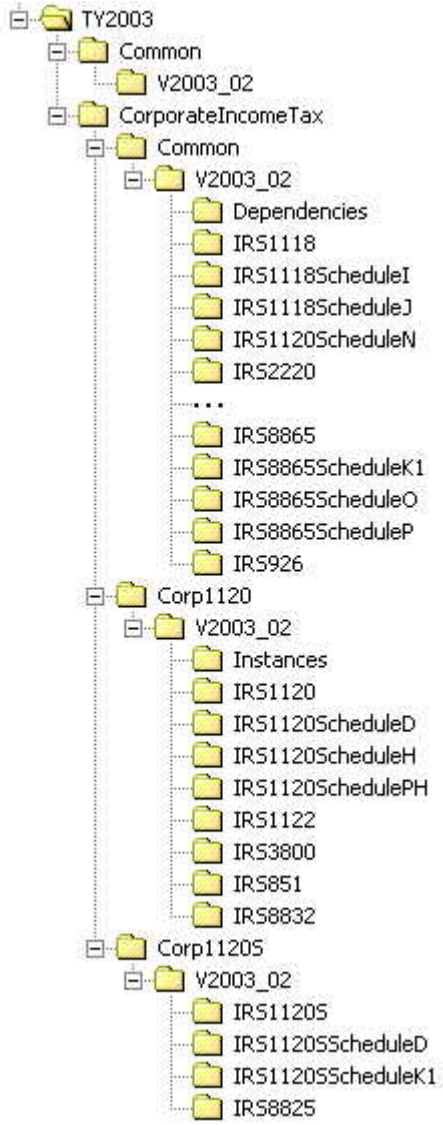


Figure 3 - Abbreviated Screenshot of Schema File Directories

Referring to Figure 3 above, here are the descriptions of the major directories found in the zip file located on the IRS.gov website:

TY2003/ - directory containing schemas for a Tax Year (TY). It simply serves as a container around return family hierarchies (**CorporateIncomeTax/** above) and schemas common across all return families (**Common/** above).

TY2003/Common/V2003_02 – contains the common schemas used by all return families (1120x, 99x, …) for a particular return version (“2003V02” in this case; specified as an attribute of the “Return” element). It contains

efileTypes.xsd, which defines most common types used by all the schemas. Also, it contains **efileMessage.xsd**, which defines the transmission header and manifest XML structures as well as the acknowledgement structure.

TY2003/CorporateIncomeTax - directory for the corporate income tax return family (1120/1120S). It simply serves as a container around 1120 return schemas (**Corp1120/** above), 1120S return schemas (**Corp1120S/** above), and schemas common across both 1120 and 1120S returns (**Common/** above).

Other return families would be parallel to this structure. For example, for Tax Exempt and Government Entities (TEGE), the directory would be **TY2003/TEGE**, and it would contain a common folder and specific return folders too.

TY2003/CorporateIncomeTax/Common/V2003_02 - directory for schemas common to 1120/1120S returns for a particular return version (2003V02 in this case). It contains **ReturnHeader1120x.xsd**, which defines the common structure of the `<ReturnHeader>` element for 1120/1120S returns.

TY2003/CorporateIncomeTax/Common/V2003_02/Dependencies - contains schemas for the supporting documents/attachments common to 1120/1120S returns. It includes the IRS Corporate Payment schema.

TY2003/CorporateIncomeTax/Corp1120/V2003_02 - directory for schemas specific to the 1120 return type for a particular return version (2003V02 in this case). It contains **Return1120.xsd**, which is the starting point for the definition of the 1120 return.

TY2003/CorporateIncomeTax/Corp1120S/V2003_02 - directory for schemas specific to the 1120S return type for a particular return version (2003V02 in this case). It contains **Return1120.xsd**, which is the starting point for the definition of the 1120S return.

TY2003/CorporateIncomeTax/Common/V2003_02/IRS*/
TY2003/CorporateIncomeTax/Corp1120/V2003_02/IRS*/
TY2003/CorporateIncomeTax/Corp1120S/V2003_02/IRS*/ - the schemas specific to a form/schedule are found below these directories, where the name of form/schedule follows the "**IRS**" prefix in the directory name. For example, schemas specific to Form 1120 would be found in the directory "**TY2003/CorporateIncomeTax/Corp1120/V2003_02/IRS1120/**". Within these "**IRS***" directories, the schema for the form/schedule is identical to the directory name (plus an "xsd" extension), and all other files are the schemas for supporting info documents specific to that form/schedule. Note that the "**IRS***" directories under the "**Common"** directory contain the form/schedule schemas common to both 1120 and 1120S return types.

The bulk of the files are the form/schedule schemas along with the common supporting info schemas. These files describe the XML elements and attributes that correspond to the fields and notations on the IRS forms. Generally, the flow of the element definitions follows the flow of the form. Most of the elements are of a type defined in **efileTypes.xsd**. If not, the type is defined within the containing schema. A typical example of a schema element definition is shown below. It defines an element, `<CostOfGoodsSold>`, which corresponds to an amount field on the form.

```
<!-- Cost of Goods Sold -->
<xsd:element name="CostOfGoodsSold" type="USAmountType" minOccurs="0">
      <xsd:annotation>
            <xsd:documentation>
                  <Description>Cost of goods sold</Description>
                  <LineNumber>2</LineNumber>
            </xsd:documentation>
      </xsd:annotation>
</xsd:element>
```

Tables on the forms are defined as repeating groups in the schemas. These repeating groups are actually made up of an element that repeats an unbounded number of times, which wraps around elements defining the column fields. A typical abbreviated example of a repeating group is shown below.

```
<xsd:complexType name="ItemizedOtherAssets">

      <!—- Itemized Other Asset (table) -->
      <xsd:element name="ItemizedOtherAsset" type="OtherAssetsType" minOccurs="0"
maxOccurs="unbounded" />

</xsd:complexType>

<!—Other Assets Info -->
<xsd:complexType name="OtherAssetsType">
      <xsd:sequence>
            <xsd:element name="Description" type="LineExplanationType"
      minOccurs="0" />
            <xsd:element name="Amount" type="USAmountType" minOccurs="0" />
      </xsd:sequence>
</xsd:complexType>
```

8. How are the supporting documents attached to forms and fields in the return data?

The supporting documents typically are of the following types:

- Forms
- Schedules
- Statements
- Elections
- Attachments
- Payment records
- Notices

The structure of each supporting document that can be included in the return is defined by the IRS in an XML Schema. Only XML documents can be "attached"; non-XML documents (e.g., the signature document) are included as a separate MIME part in the multi-part structure of the return. In the Modernized e-File system, when a document is "attached" to an (XML) element then there is a reference from the element to the attached document.

The supporting documents can be "attached" to a line or to a form/schedule, or the return.

When a supporting document is attached to a line there is a reference from the line (i.e., the element that represents the line) to the attached document.

When a supporting document is attached to a form/schedule there is a reference from the form/schedule (i.e., the root element of the form/schedule) to the attached document.

When a supporting document is attached to the return it is just included within the return per the content model of the return; there is NO reference to it in this case. For example, when the IRS Payment Record (XML element name "IRSCorporatePayment") is "attached" to the return, it is just included within the ReturnData element of the return; there is no reference to it from any line or from any form/schedule.

Two attributes have been defined to indicate the "attachment" of a return document to an XML element. The attribute named "referenceDocumentName", wherever it appears, is used to enumerate the document(s) that can be attached to the element. The attribute named "referenceDocumentId" establishes a reference from the element where it appears, to the attached document(s).

8.1. Attaching XML documents to forms and fields in a return

Here is a simple example of how the supporting document named "ItemizedOtherIncomeSchedule" is attached to line 10 on form 1120.

The schema for form 1120 (IRS1120.xsd) specifies that you can attach the document named "ItemizedOtherIncomeSchedule" to line 10. The name of the element that represents line 10 is "OtherIncome". Here is how the document is "attached" (create a reference) to the element that represents line 10:

```
...
<OtherIncome referenceDocumentId="DEPa1">8880</OtherIncome>
...
```

The referenceDocumentId attribute contains a reference to (or points to) a document whose documentId is "**DEPa1**". In this example it is the documentId of the "ItemizedOtherIncomeSchedule" document as indicated below:

```
<ItemizedOtherIncomeSchedule documentId="DEPa1">
  <ItemizedIncome>
   ...
  </ItemizedIncome>
</ItemizedOtherIncomeSchedule>
```

Each return document must have a unique documentId within a return.

8.2. Sample return with XML documents as attachments

```
<xml version="1.0" encoding="UTF-8"?>
<Return returnVersion="2002V01">
<ReturnHeader binaryAttachmentCount="0" subsidiaryReturnCount="0">
  <ReturnId>01000020020860000001</ReturnId>
  <Timestamp>2002-03-26T13:00:05-05:00</Timestamp>
    ...
    ...
</ReturnHeader>
<ReturnData documentCount="22">

<!—Four documents are "attached" in this example- three to the form 1120, and one
to the OtherIncome element (which represents line 10 on form 1120). This means
that there are three references from the root element of Form 1120 to three return
documents that are attached to the form; and there is one reference from line 10
to the attached return document. -->

<!—here is how the three documents "StockOwnershipForeignCorpStmt",
"DualConsolidatedLossesStmt", and another instance of "DualConsolidatedLossesStmt"
are attached to form 1120 →

  <IRS1120 documentId="DOC0001" referenceDocumentId="DEPc1 DEPd1 DEPd2" >
    ...
    ...
     <!-- document "ItemizedOtherIncomeSchedule" is attached to the element "OtherIncome"-->
    <OtherIncome referenceDocumentId="DEPa1">8870</OtherIncome>
    ...
    ...
  </IRS1120>
  ...
  ...
  <!-- Supporting document attached to an element in Form 1120 -->
  <ItemizedOtherIncomeSchedule documentId="DEPa1">
    <ItemizedIncome>
     ...
     ...
    </ItemizedIncome>
  </ItemizedOtherIncomeSchedule>
  ...
  ...
  <!-- Supporting document attached to Form 1120 -->
  <StockOwnershipForeignCorpStmt documentId="DEPc1">
    ...
```

```
      ...
   </StockOwnershipForeignCorpStmt>
   ...
   ...
   <!--  Supporting  documents attached to Form 1120 -->
   <DualConsolidatedLossesStmt documentId="DEPd1">
      <Statement>Statement One</Statement>
   </DualConsolidatedLossesStmt>

   <DualConsolidatedLossesStmt documentId="DEPd2">
      <Statement>Second Statement</Statement>
   </DualConsolidatedLossesStmt>
   ...
   ...
</ReturnData>
</Return>
```

8.3. Attaching non-XML documents, i.e., binary (PDF)

Non-XML documents, also referred to as binary attachments, can be “attached” to the return, and also “attached” to forms and fields in the return. Attaching to the return is the same as “including” in the return as explained in the beginning of this section. Attaching to forms and fields in the return is the same as “referencing” from particular XML elements in the return as explained in the beginning of this section. It is important to note that this binary attachment referencing feature depends on specific reference point definitions within the schemas, and that, currently, there are no such definitions in place. It is understood that the IRS may define such reference points at some point in the future. Until then, simply including binary attachments (without referencing) in the return will continue to be the minimal way to attach non-XML documents. In both scenarios, the attribute binaryAttachmentCount of element ReturnHeader is used to indicate the number of binary attachments included in the return.

8.3.1. Including binary attachments with a return

As previously described and depicted, binary attachments must be placed in separate MIME parts within the MIME multi-part of a return, following the XML part of the return. These simple placements are considered “attachments” to the return. Also, as previously described, their “Content-Location” MIME headers must be unique, to ensure proper identification. Furthermore, it is imperative that their “`ContentDescription:`” MIME headers contain non-empty values, in order to provide useful descriptions of the attachments.

8.3.2. Referencing binary attachments from forms and fields in a return

Again, the referencing of binary attachments from elements within the XML return, is a feature that depends on the definitions of reference points within the schemas. Currently, there are no such definitions in place. This subsection describes these definitions and others that support them.

To reference the binary attachments from the XML return, the elements from where references are made will contain the attribute definition pair: `referenceDocumentId` (of IdListType) and `referenceDocumentName=”BinaryAttachment”`. Like in the aforementioned XML document referencing, this `referenceDocumentId` will map to a `documentId` of an XML document that describes the binary document. The following is an example of the XML document that describes the binary attachment:

```
<BinaryAttachment documentId=”00000050”>
   <DocumentType>PDF</DocumentType>
   <Description>Certificate of Merger</Description>
   <AttachmentLocation>PDF0001</AttachmentLocation>
```

```
</BinaryAttachment>
```

The DocumentType will be of an enumerated type of string values, identifying which types of binary attachments are allowed. The Description will provide a brief description of the attachment, and will override the value of the attachment's "Content-Description:" MIME header. Furthermore, the AttachmentLocation would contain the "Content-Location:" MIME header value of the corresponding binary attachment. This is how the binary MIME part is referenced.

8.4. Segment of a transmission file sample showing an individual return with its binary attachment with reference from a particular XML location

The transmission sample below shows a binary attachment (PDF) named "Certificate of Merger", which is attached to Form 4562. It shows the reference from the "IRS4562" form element to the XML document, "BinaryAttachment", which in turn, refers to the corresponding MIME location via its "AttachmentLocation" element.

```
--MIME1120Boundary
Content-Type: Multipart/Related; boundary=MIME1120Return0001Boundary;
type=text/xml
Content-Location: 01000020020860000001

--MIME1120Return0001Boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-Location: DataPart001

<Return returnVersion="2002V01">
   <ReturnHeader binaryAttachmentCount="1" subsidiaryReturnCount="0">
      ...
   </ReturnHeader>
   <ReturnData documentCount="11">
      <IRS1120 documentId="DOC0001">
         ...
         </IRS1120>        ...
      <IRS4562 documentId="DOC0005" referenceDocumentId=”DOC0011”>
         ...
      </IRS4562>
         ...
      <BinaryAttachment documentId=”DOC0011”>
         <DocumentType>PDF</DocumentType>
         <Description>Certificate of Merger</Description>
         <AttachmentLocation>PDF0001</AttachmentLocation>
</BinaryAttachment> </ReturnData>
</Return>

--MIME1120Return0001Boundary
Content-Type: application/pdf
Content-Transfer-Encoding: Binary
Content-Location: PDF0001
Content-Description: Certificate of Merger
...
...

--MIME1120Return0001Boundary--
```

```
--MIME1120Boundary
```

9. Guidelines for composing the return data structure

This section lists some specific guidelines for composing returns, based on popular errors encountered during previous production releases. Mostly, MIME format errors dominate the problems found.

MIME requirements to follow:

- The preceding hyphens "—" (2 dashes) are required in the beginning of MIME boundary
- The MIME boundary cannot be prematurely or improperly ended
- The value specified for the "type" attribute, in the "Content-Type" MIME Header, must be enclosed within double quotes (i.e. `type="text/xml"`)
- No blank line should be present following/after the declaration of the beginning of a MIME boundary
- A blank line is required to be present right before the declaration of the beginning of a MIME boundary and before the XML declaration of SOAP envelope or return(s), in order for the XML document root element to be properly detected

  NOTE: In same instances, the data might already have a **CR and LF** (Hex pattern: **0D 0A**) before the "—" (2 hyphens) where a MIME boundary starts, but error(s) can still be thrown due to "Missing start boundary". In such instances, the proper Hex pattern needed to correct the problem is: **0D 0A 0D 0A**

  The same Hex pattern **0D 0A 0D 0A** should also be applied before the XML declaration, in order to avoid SAXParserException: "Document root element is missing"

- The Hex pattern: **0D 0A** (**CR** and **LF**), is required before/after the declaration of each MIME Headers, specifically the following headers:

  Content-Type
  Content-Transfer-Encoding
  Content-Location

XML return problems:

- There should be no carriage return, nor line, nor space, right before the end tag </Reference> in the Transmission Manifest. XML validation will interpret such whitespace as text, when the element should be empty

  Recommendations:
  ```
  <Reference ...></Reference>
  <Reference .../>
  ```

- The namespace prefix "efile" must be defined in the return's root element <Return …>.

  The format is:
  ```
  xmlns:efile="http://www.irs.gov/efile"
  ```

10. How do I validate my return against the XML schemas?

Here is a high-level content model of a Corporate `Return` XML document:

```
<xml version="1.0" encoding="UTF-8"?>
```

```
<!-- Return1120.xsd for and 1120 Return -->
<Return returnVersion="2002V01">

   <!-- ReturnHeader.xsd for Return Header -->
   <ReturnHeader binaryAttachmentCount="0" subsidiaryReturnCount="0">
   </ReturnHeader>

   <!-- ReturnData1120.xsd or ReturnData1120S.xsd for All Forms/Fields, schedules
and XML Attachments -->
   <ReturnData documentCount="22">
      <!—- IRS1120.xsd, XML Schema for Form IRS 1120 -->
      <IRS1120 documentId="DOC0001">
         ...
         <!-- Form Fields -->
         <OtherIncome referenceDocumentId="DEPa1">8770</OtherIncome>
         ...
      </IRS1120>
      ...
      <!—- XML Schema for a Dependency, Itemized Other Income Schedule-->
      <ItemizedOtherIncomeSchedule documentId="DEPa1">
         ...
      </ItemizedOtherIncomeSchedule>
      ...
   </ReturnData>

</Return>
```

Here is a brief description of above content model:

- The **`Return`** is the root element in each return. The file named Return<TaxType>.xsd (where <TaxType> is one of these values- 1120, 1120S, 990, 990EZ,1120POL, 8868) defines the content model of a return. The element **Return** is defined in this file.
- The **`Return`** comprises of two mandatory child elements, **`ReturnHeader`** and **`ReturnData`**. In general, each return has its own ReturnHeader and ReturnData elements that are specific to that return. However, the ReturnHeader for an 1120 and an 1120S return is the same, and is defined in ReturnHeader1120x.xsd file. Similarly, the ReturnHeader for a 990 return and a 990-EZ return is also the same, and is defined in ReturnHeader99x.xsd file.
- A **`ReturnHeader`** contains common elements of a return, e.g., taxpayer name, EIN, address, etc.
- Each individual return document is either a form or schedule or supporting material and has a separate XML schema defined for it.
- The **ReturnData** element contains all return documents (forms, schedules and supporting materials) for a return.

Below are some XML resources regarding XML schemas and software tools and parsers (these resources are provided for information only—the IRS is not endorsing any product)

- W3C XML Home Page: `http://www.w3.org/XML/`
- W3C XML Schema Home Page: `http://www.w3.org/XML/Schema`
- XML Spy: `http://www.xmlspy.com/`
- Apache Xerces parser toolkit: `http://xml.apache.org/`
- Microsoft MSDN library: http://www.microsoft.com/xml
- You may chose any third party parser toolkit or use your own.

10.1. Validating a single return document

As stated earlier, each individual return document (form, schedule, supporting material etc) has its own XML schema defined. Therefore, one does not need to compose the whole return (XML document e.g., per `Return1120.xsd` for an 1120 return) in order to validate that singe return document (a form, schedule, a supporting material).

In other words, one may simply assemble data pertaining to that individual return document (a form, schedule, supporting material) and immediately validate it using its own XML schema.

Here is an example of how to validate a single Return Document, "Itemized Other Income Schedule" XML document.

- Assemble all the elements needed for the `ItemizedOtherIncomeSchedule.xsd` document as follows:

```
<ItemizedOtherIncomeSchedule documentId="ABC00010" >
  <ItemizedIncome>
     <IncomeType>Trade</IncomeType>
     <Amount>22330</Amount>
     ...
     <PartnershipAmount>11120</PartnershipAmount>
  </ItemizedIncome>
</ItemizedOtherIncomeSchedule>
```

- Add the following to the above (text and other required attributes to the root which are shown in **bold** in the example below) to make a stand-alone XML document point to appropriate XML schema file. Please realize that you need to specify the directory where the **`ItemizedOtherIncomeSchedule.xsd`** `file resides on your machine.` Here is a sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<ItemizedOtherIncomeSchedule xmlns="http://www.irs.gov/efile"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.irs.gov/efile
TY2002/CorporateIncomeTax/Corp1120/V2002_01/IRS1120/ItemizedOtherIncomeSched
ule.xsd" documentId="ABC00010">
  <ItemizedIncome>
     <IncomeType>Trade</IncomeType>
     <Amount>22330</Amount>
     ...
     <PartnershipAmount>11120</PartnershipAmount>
  </ItemizedIncome>
</ItemizedOtherIncomeSchedule>
```

Now, validate the above XML document using your own XML parsers/validators.

10.2. Validating the whole return

Following are the two ways one could validate a return against the xml schemas.

(1) One-step approach

Prepare all individual XML documents for the return (e.g., ReturnHeader, forms, schedules, and supporting materials, etc.), and then assemble and validate against the appropriate schema, for example, validate an 1120 return against the schema specified for the 1120 return in the `Return1120.xsd` file.

(2) Multi-step approach

There are several different ways one could do multi-step validation of the return. Here is one way:

- Prepare a `ReturnHeader` document and validate against its schema, `ReturnHeader1120x.xsd`.
- Prepare each individual return document (forms, schedules, and supporting materials) and validate against its schema.
- Assemble all the return documents into `ReturnData` and validate against the ReturnData schema of the return e.g., `ReturnData1120.xsd` for an `1120 return data.`
- Assemble `ReturnHeader` and `ReturnData` into `Return` and validate against the appropriate schema e.g., `Return1120.xsd` for an 1120 return.
- This completes the whole return process validation.

Here is another way which is a variation from the procedure shown above:

- Prepare a `ReturnHeader` document. One may or may not validate against its schema, `ReturnHeader1120x.xsd` before proceeding.
- Prepare each individual return document (Forms, schedules, and supporting materials). One may or may not validate against its schema before proceeding.
- Assemble all the return documents into `ReturnData`. One may or may not validate against the respective schemas, `ReturnData1120.xsd` or `ReturnData1120S.xsd` before proceeding.
- Assemble `ReturnHeader` and `ReturnData` into `Return` and validate against the appropriate schema e.g., validate an 1120 return against the schema for an 1120 return defined in the `Return1120.xsd` file.
- This completes the whole return validation process.

Choose the procedure that is convenient and best fits your needs.

10.3. Validating the transmission envelope including its contents

The transmission file is a MIME multi-part document that conforms to "SOAP 1.1 with attachments" standard. The first part of the multi-part document is the SOAP envelope and remaining parts are SOAP attachments. MIME boundaries separate the parts in the multi-part document. The SOAP envelope maintains transmission level information, and SOAP attachments are the returns included in the transmission file.

The SOAP envelope consists of a SOAP header and a SOAP body. The SOAP header, also referred to as the *transmission header* in the Modernized e-File system, contains information about the transmitter and the transmission. The SOAP body, also referred to as the *transmission manifest*, contains a list of all SOAP attachments in the transmission file.

Validation of SOAP envelope or transmission envelope including its contents consists of the following steps.

- Validate the SOAP envelope or transmission envelope XML instance against the SOAP schema, SOAP.xsd. The standard SOAP schema has been used without modification, http://schemas.xmlsoap.org/soap/envelope/.

- Transmission header SOAP structure must consist of a single element, `TransmissionHeader`. Validate `TransmissionHeader` element against the schema for the TransmissionHeader defined in `efileMessage.xsd` file.

  See section “Validating a single return document” for details on how to validate an individual XML document (in this case, transmission Header XML document).

- Transmission body SOAP structure must consist of a single element, `TransmissionManifest`. Validate `TransmissionManifest` element against the schema for the TransmissionManifest defined in `efileMessage.xsd` file.

  See section "Validating a single return document" for details on how to validate an individual XML document (in this case, transmission manifest XML document).

10.3.1. Sample SOAP envelope with TransmissionHeader and TransmissionManifest

Below is a sample SOAP Envelope based on `SOAP.xsd`. Note that the contents for the `SOAP:Header` and `SOAP:Body` are based on the `TransmissionHeaderType` and `TransmissionManisfestType` defined in `efileMessage.xsd`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!– SOAP:Envelope must be validated against SOAP.xsd -->
<SOAP:Envelope xmlns="http://www.irs.gov/efile"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:efile="http://www.irs.gov/efile"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ SOAP.xsd
http://www.irs.gov/efile efileMessage.xsd">
  <SOAP:Header>
     <!– efile:TransmissionHeader must be validated against efileMessage.xsd -->
    <efile:TransmissionHeader transmissionVersion="2002V01">
      <TransmissionId>TID01</TransmissionId>
      <Timestamp>2002-03-27T14:30:00-05:00</Timestamp>
      <Transmitter>
        <ETIN>24317</ETIN>
      </Transmitter>
      <ProcessType>P</ProcessType>
    </efile:TransmissionHeader>
  </SOAP:Header>
  <SOAP:Body>
     <!– efile:TransmissionManifest must be validated against efileMessage.xsd --
>
    <efile:TransmissionManifest count="4">
      <Reference contentLocation="01000020020860000001"
electronicPostmark="2002-03-27T08:20:00-05:00" />
      <Reference contentLocation="01000020020860000002"
electronicPostmark="2002-03-27T08:20:10-05:00" />
      <Reference contentLocation="01000020020860000003"
electronicPostmark="2002-03-27T08:20:20-05:00" />
      <Reference contentLocation="01000020020860000004"
electronicPostmark="2002-03-27T08:20:30-05:00" />
    </efile:TransmissionManifest>
  </SOAP:Body>
</SOAP:Envelope>
```

11. How are errors reported?

The structure and content of the transmission file and each Return/Extension included in the transmission file is validated to ensure that it conforms to the structure published by the IRS and the rules established by the IRS. The structure of the transmission file is checked for conformance to MIME standard and the structure of the return/extension data is checked to ensure that it conforms to the XML Schemas published by the IRS. The return/extension data is validated against the IRS databases and checked for conformance to business rules published by the IRS. When either the structural violations are discovered, or the data fails some business rules, errors are generated and reported back in an Acknowledgement file.

The MIME-Headers in the transmission file are validated to ensure that their values (and their parameter values, if any) are set correctly. For example the MIME Header "X-eFileRoutingCode" must be present and its value must be one of the three allowed values - 'MEF', '94X', 'PINREGISTRATION'. In the case of Modernized e-File, the value would always be “MEF”. Similarly, the content of the transmission envelope is validated to ensure that it is structurally correct (per SOAP1.1) and each reference in the transmission manifest is found in the transmission file. If the transmission file structure violates some MIME rules, and/or business rules published by the IRS, the whole transmission file is rejected; the returns/extensions included within the transmission file are NOT checked for errors in this case.

If the transmission file structure conforms to the MIME standard and IRS business rules, then each return/extension in the transmission file is validated to make sure that the data is structurally correct and conforms to the published business rules. Structural correctness means that the data conforms to the published XML Schemas. For example, all required elements are present, and they conform to their established cardinality. Conforming to business rules means that the relationships amongst the data elements hold as stated in the published business rules. For example, the filer is established in the IRS databases. When either the structural violations are discovered, or the data fails some business rules, errors are generated and reported back in an Acknowledgement file.

Three kinds of data validation is performed by the Modernized e-File system-

- Structural Validation- conformance of XML data against the published Schemas, and conformance of the file structure to the MIME standard
- Database Validation- conformance of data with controls established in the IRS databases
- Business Rule Validation- conformance of data to the established relationships amongst data elements

When a structural violation is discovered in the transmission file, the whole transmission file is rejected. For example, if the ID of the transmitter, ETIN, is not included in the transmission header, the transmission file is rejected, and the contents of the file are not examined.

When a structural violation is discovered in a return/extension, the return/extension is rejected. The section titled “How are the structural errors reported?” provides details of the error structure reported for such errors.

When the data violates a business rule (e.g., the EFIN provided is not listed in the IRS database), or when the data violates a business rule that checks for data consistency (e.g., on Form 1120, Item D “Total Assets” must equal Schedule L line 15d), then depending on the severity of the rule, the return/extension may be rejected.

11.1. Severity of the business rules and its implications

Each business rule is assigned a severity by the IRS that determines if a return is accepted or rejected. There are three severity levels-

1. Reject And Stop
2. Reject
3. Alert

When a business rule with a severity of 'Reject And Stop' is violated, the return/extension is rejected, an error is reported in the Acknowledgement file, and no further processing of the data is done. In this case, the Acknowledgement file may not contain all errors that may exist in the return/extension data.

When a business rule with a severity of 'Reject' is violated, the return/extension is rejected, an error is reported in the Acknowledgement file, and the system continues to find other errors, if any. In this case, the Acknowledgement file contains either all errors discovered by the system (if the number of errors is less than the IRS established threshold), or the number of errors equal to the threshold value.

When a business rule with a severity of 'Alert' is violated, an error is reported in the Acknowledgement file and the system continues to find other errors, if any. The business rules with severity 'Alert' do not cause the return/extension to be rejected.

11.2. Error Structure

Each error generated contains the following information:

- **Path** - (Xpath) to the data element causing the violation, when available
- **Error Category** - Errors are grouped into a small number of categories
- **Error Message** - Rule text or XML validator message
- **Rule Number** - Each rule is identified by a unique rule number.
- **Severity** - 'Reject And Stop', 'Reject' or 'Alert'
- **Data value** - Data value causing the violation - when appropriate

Errors are reported back to the sender in an acknowledgement file. The structure of the error element in the acknowledgement file is as given below:

```
<Error>
        <Xpath></Xpath>
        <ErrorCategory></ErrorCategory>
        <ErrorMessage></ErrorMessage>
        <RuleNumber></RuleNumber>
        <Severity></Severity>
        <DataValue></DataValue>
</Error>
```

Example below shows a section of the acknowledgement file when the following business rule is violated:

Rule Number: F1120-035
Rule Text: If Form 1120, Schedule J, Line 6e has a non-zero value, then Form 8827 must be attached.
Severity: Reject

Acknowledgement File

```
<ReturnAcknowledgement xmlns="http://www.irs.gov/efile"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.irs.gov/efile ../message/efileMessage.xsd">
 <ReturnId>01000020020860000017</ReturnId>
 <Errors errorCount=2 >
  <Error errorId="1">
    <Xpath>/Return/ReturnData/IRS1120/IRS1120ScheduleJ/PriorYearMinimumTaxCredit</Xpath>
    <ErrorCategory>Missing Document</ErrorCategory>
    <ErrorMessage> If Form 1120, Schedule J, Line 6e has a non-zero value, then
            Form 8827 must be attached.
    </ErrorMessage>
```

```
    <RuleNumber>F1120-035</RuleNumber>
    <Severity> Reject </Severity>
    <DataValue></DataValue>
  </Error>
  <Error errorId="2">
        .
  </Error>
</Errors>
 .
 </ReturnAcknowledgement>
```

11.3. How are the structural errors reported ?

When the transmission file violates MIME structure established by this document per MIME standard, or when the return/extension data does not conform to the XML Schemas, structural errors are generated. Structural errors are reported using a category named “XML Error”.

- **XML Error**
  When the data violates the Schema (Content Model) specification (e.g., missing a required element, or multiple occurrences of an element when only one is allowed, e.g., Form 1120),or when the Transmission File structure does not conform to the MIME structure established by this document.

The example below shows the structure of the error message when the return data does not conform to the schema. Here an 'XML Error' is generated because a required element, "City", was not provided as part of the preparer firm's address. Please note that the xpath points to the next data element, "State" when "City" is not provided.

Business Rule

> The return (XML documents) must conform to the version of the XML Schema indicated by the 'version' element in the return header.

This business rule covers all XML structural errors that are not specifically inspected by the Modernized e-File system. When violated, the following data is reported in the Acknowledgement file:

Xpath: /Return/ReturnHeader/PreparerFirm/PreparerFirmAddress/State

Error Category: XML Error

Error Message: The return (XML documents) must conform to the version of the XML Schema indicated by the 'version' element in the return header.
{//COMPLEX_E_UNEXPECTED_CONTENT// Unexpected Content "State"; expected "City".}

Rule Number: X0000-005

Severity: Reject And Stop

Data Value:

Text in *italics* is provided by the XML Validator (presently XML Validator by TIBCO is used)

11.4. How are the data errors reported ?

When the transmission file contains invalid values for the MIME headers (or their parameters) or when the return/extension data violates one or more business rules, data errors are generated. Examples of errors of this kind include missing supporting information, violation of certain relationships amongst data elements, and mismatch of data against IRS databases. Errors of this nature are classified into the following categories:

- **Database Validation Error**
  When some data provided in the return must be present in the IRS database, but is not (e.g., SoftwareID in the Return Header must have passed testing for the return type and tax year).
- **Missing Document**
  When a return document is required per some business rule and is not included in the return (e.g., If Form 8586 Line 4 has a non-zero value then one or more Form 8609 must be attached to the return).
- **Multiple Documents**
  When more than the allowable number of documents (per some business rule) are included in the return (e.g., If Form 1120S, Schedule K, Line 12c has a non-zero value, then no more than one Form 3468 must be attached)
- **Missing Data**
  When data that should have been provided per some business rule is not provided (e.g., If Form 5471 is attached, then Schedule N (Form 1120) Line 4b must have a non-zero value).
- **Incorrect Data**
  When data is syntactically correct, but violates some business rule (e.g., Form 6478, Line 1(a), cannot be greater than 15 million).
- **Data Mismatch**
  When the value in two fields should be the same (may be the source is different), but is not (e.g., Form 1120, Line 2 must equal Form 1120 Schedule A Line 8).
- **Duplicate Condition**
  When a return or transmission is a duplicate of a previously accepted return or transmission.
- **Math Error**
  When Form 1120, Line 31[TotalTax] plus(+) Line 33[EstimatedTaxPenalty] is less than Line 32h[TotalPayments], then Line 32h minus (-) [Line 31 plus(+) Line 33] must equal Line 35[OverpaymentAmount].
- **Not On Time**
  When a return/extension is received after the due date.
- **Information Message**
  When some condition in the data causes an informational message to be sent to the filer (e.g., If Form 990, Line 77 checkbox "Yes" is checked, then a "confirmed copy of changes" document must be sent to IRS.)
- **Unsupported**
  When a submitted item (form, feature, format of something etc.) is sent to a location that does not accept it, or an unusual condition is encountered in data e.g., The Transmission File must be free of virus. A virus was found in this file.

The Error structure remains the same as described for schema validation. The 'Error Message' in this case is the text of the Business rule as shown by the example below:

The example below illustrates the error structure when a 'Data Mismatch' error is encountered:

Business rule If Form 1120, Line 4 has a non-zero value, then Form 1120, Line 4 must equal Form 1120, Schedule C, Line 19.

Form 1120, Line 4 = 12345, Form 1120, Schedule C, Line 19 = 22345

When this business rule is violated, the following data is reported in the Acknowledgement file:

Xpath: /Return/ReturnData/IRS1120/Dividends

Error Category: Data Mismatch

Error Message: If Form 1120, Line 4 has a non-zero value, then Form 1120, Line 4 must equal Form 1120, Schedule C, Line 19

Rule Number: F1120-020

Severity: Reject

Data Value: 12345

12. Identifying numbers and their scope

Within the schemas, identifying numbers are defined at the transmission, return, and form levels. These identifying numbers are considered to be key elements/attributes in their containing structures. Their purpose is to identify unique entities, such as a document, organization, person, etc.

Transmission Level Identifying Numbers:

Transmission ID – element <TransmissionId> in the <TransmissionHeader> element definition, identifies a unique transmission for the tax year. It is created by the transmitter. It can be up to 30 characters in length, is alphanumeric, and can contain characters ":", ".", and "-". This pattern allows the transmitter to use a timestamp within the field if desired. This identifying number is also found in the <TransmissionAcknowledgement> element definition.

Transmitter's ETIN – found within the <TransmissionHeader> element definition, identifies the unique electronic transmitter. It's a unique 5 digit identification number assigned by the IRS.

Return Level Identifying Numbers:

Return ID – element <ReturnId> in the <ReturnHeader> element definition, identifies a unique return within the transmission. It must be unique within a tax year. It is assigned by the originator. It is 20 characters in length, and follows the pattern is: EFIN (6 digits) || Year (4 digits) || Julian Day (3 digits) || Sequence Number (7 alphanumeric). The first six positions of the ReturnID must match the originator's EFIN. This identifying number is also found in the <ReturnAcknowledgement> element definition.

Software ID – element <SoftwareId> in the <ReturnHeader> element definition, identifies the software used to compose the return. It's an 8 digit numeric field assigned by the IRS.

Originator's EFIN – found within the <ReturnHeader> element definition, this is the originator's Electronic Filing Identification Number. It's a 6 digit numeric field, where the first 2 digits represent a pre-defined IRS district office code. This identifier is assigned by the IRS.

Filer EIN – found within the <ReturnHeader> element definition, it is the Employer Identification Number of the business for which the return is being filed. It's a 9 digit numeric field, where the first 2 digits represent a pre-defined IRS district office code. This identifier is assigned by the IRS.

Preparer's SSN or PTIN – found within the <ReturnHeader> element definition, this is a choice between a person's Social Security Number or Preparer Personal Identification Number. SSN is a 9 digit numeric field, and PTIN is 9 digits, beginning with the letter 'P' followed by 8 numeric digits.

Preparer Firm's EIN – found within the <ReturnHeader> element definition, this is the Employer Identification Number of the firm which prepared the return (if applicable). It is a 9 digit numeric field, where the first 2 digits represent a pre-defined IRS district office code.

Form Level Identifying Numbers:

Software ID – attribute softwareId in every top level element in the form/schedule schemas, this number identifies the software used to compse the single form/schedule XML instance. It is an 8 digit numeric field assigned by the IRS. If more than one software package is used to compose the return, identify the software ID used to create each form within the return.

Document ID – attribute documentId in every top level element in the form/schedule schemas, uniquely identifies a single form/schedule XML instance within the return. It can be up to 30 characters

in length, and is alphanumeric, and can contain characters colon (":"), period ("."), and hyphen ("-"). This pattern should allow for a timestamp to be used within the field. Its value is assigned by the ERO's software.

Reference Document ID – attribute `referenceDocumentId` found in element definitions where attachments of supporting info documents are made, refers to a unique form/schedule XML instance (identified by its `documentId` attribute) within the return. Thus, this attribute's structure is identical to the structure of the `documentId` attribute.

Reference Attachment ID – attribute `referenceAttachmentId` found in element definitions where attachments of binary documents are made, refers to a unique binary attachment (identified by a `Content-Location` header value) within the MIME multi-parts of a return. This attribute's structure is identical to the structure of the `documentId` attribute.

Note: References to binary attachments from within the return data are not supported in Release 1. Hence this attribute is not supported in Release 1.

Reference Attachment Name – attribute `referenceAttachmentName` found in element definitions enumerates the names of binary attachments that can be attached to the element where this attribute appears.

Note: References to binary attachments from within the return data are not supported in Release 1. Hence this attribute is not supported in Release 1.